A two stage learning algorithm for a Growing-Pruning Spiking Neural Network for pattern classification problems

A Two Stage Learning Algorithm for a Growing-Pruning Spiking Neural Network for Pattern Classification Problems

Shirin Dora
School of Computer Engineering,
Nanyang Technological University,
Singapore-639798.
Email: shirin002@ntu.edu.sg

Suresh Sundaram
School of Computer Engineering,
Nanyang Technological University.
Singapore-639798.
Email: ssundaram@ntu.edu.sg

Narasimhan Sundararajan
School of Electrical and,
Electronic Engineering,
Nanyang Technological University.
Singapore-639798.
Email: ensundara@ntu.edu.sg

***Abstract*—This paper presents a two stage learning algorithm for a Growing-Pruning Spiking Neural Network (GPSNN) for pattern classification problems. The GPSNN uses three layered network architecture with input layer employing a modified population coding and, leaky integrate-and-fire spiking neurons in the hidden and output layers. The class label for a sample is determined according to the output neuron with minimum spike latency. The learning algorithm for the GPSNN employs a two stage learning mechanism. In the first stage, the hidden layer is grown and adapted to map the inputs to a hyperdimensional space. In the second stage, the hidden layer neurons with low dominance are pruned and the response of the most dominant neurons is mapped to the output space. The proposed approach has been evaluated on benchmark data sets from the UCI machine learning repository and the results were compared with batch as well as online spiking neural networks. The results clearly highlight that the GPSNN can achieve better performances using a compact network structure.**

I. Introduction

An Artificial Neural Network (ANN) is a highly simplified model of brain created with the aim of replicating the intelligence of humans in machines. The developments in the area of ANNs have always been motivated by improvements in our understanding of the brain. The initial belief in the neuroscience community that biological neurons encode information using the frequency of spikes led to the development of first and second generation of ANNs. However, recent experiments by neuroscientists have shown that biological neurons encode information using the precise time of spikes [1], [2], [3]. This led to the development of the third generation of ANNs known as spiking neural networks.

The first supervised learning algorithm for spiking neural networks, called SpikeProp, has been proposed in [4]. SpikeProp assumes that the Post-Synaptic Potential (PSP) of a neuron is linearly continuous around the time of spike, thereby allowing calculation of gradient. It uses a fixed network architecture of neurons modeled using spike response function and each neuron is allowed to spike only once during the simulation interval. If a neuron fails to spike during the learning process, then no gradient calculation is possible, resulting in an obsolete neuron. This problem is called 'silent neuron' problem and has been discussed in [5]. There exist other error functions in literature which don't suffer from the 'silent neuron' problem [6], [7]. It is well-known that the convergence of gradient descent based approaches is strongly dependent on the chosen parameters. In literature, other learning algorithms have been proposed which employ a Spike Timing Dependent Plasticity (STDP) based learning rule [8] and a combination of STDP and Bienenstock-Cooper-Munroe learning rule [9].

All the above algorithms require that the network architecture be fixed beforehand. But, for many real world problems, it is difficult to choose suitable network architecture without comprehensive knowledge about the input space. This motivated researchers to develop learning algorithm which could evolve the network architecture for spiking neural networks. Recently, several learning algorithms with an evolving network architecture have been proposed for spiking neural networks in [10], [11].

In [10], an evolving learning algorithm for a neural network with four layers of 2-dimensional neuronal maps has been proposed. The neurons are modeled using a simplified integrate-and-fire model and the weights for new neurons are initialized using rank order learning [12]. Rank order learning is a biologically motivated technique based on the assumption that most of the information is exchanged between neurons in earliest spikes [12], [13]. This makes rank order learning an effective technique for pattern recognition in one-shot learning mode. The new neurons are added to the network according to a similarity measure based on the Euclidean distance without taking into account the precise time of spikes. This may lead to the addition of more neurons. Further, [10] has been proposed for handling visual pattern recognition problems and can not be used as it is for problems with real valued data.

In [11], an Online Spiking Neural Network (OSNN) with a three layered architecture, has been proposed for handling problems with numerical data. It employs population coding in the input layer to convert the real valued data into spikes [4] and, spiking neurons in hidden and output layer are modeled using the spike response model. Since population coding employs multiple delays for each receptive field, the number

of parameters is drastically increased, leading to increased computational complexity. The predicted class for a sample is determined according to the output neuron with maximum frequency of spikes. The number of neurons in the hidden layer is evolved according to a clustering approach and the weights for new neurons are initialized by scaling existing neuron weights. Due to scaled weights, the new neuron also fires for already learned samples, which can lead to higher misclassification. Further, the weight adaptation is done using a Hebbian like learning rule which is not suitable for one-shot learning mode. The weights for output layer neurons are estimated using STDP and anti-STDP which limits the learning capability of the network [7], [9]. This motivated us to develop a new learning algorithm which does not have the drawbacks of STDP and utilizes the class-specific information present in the network to improve classification [14].

In this paper, a two stage Growing-Pruning Spiking Neural Network (GPSNN) has been proposed for handling pattern classification problems with real valued data. It employs a modified population coding scheme (similar to the encoding technique employed in [15], [16]) in the input layer to convert real valued data into a spike pattern. The modified population coding scheme does not require delays which results in considerably lesser parameters, hence reducing the computational effort needed. The hidden and output layer neurons are modeled as leaky integrate-and-fire neuron [17] and the predicted class is determined according to the output neuron with minimum spike latency.

The proposed learning algorithm for the GPSNN employs a two stage learning approach. In the first stage, called structure learning, the learning algorithm maps the input data to a hyperdimensional space by adding or adapting the parameters of the hidden layer neurons. The criteria for neuron addition uses a temporal similarity measure and the class-specific information, which provide an accurate estimation of the information present in the sample with respect to the network. The weights for the new neuron are initialized using an improved rank order learning technique wherein the receptive fields of any given feature are ranked individually. This improves the separability of the input patterns. Different from [10], the threshold for a newly added neuron is initialized using its PSP at a precise time instant, called ideal firing time which enables us to use a temporal measure for estimating similarity. If the criterion for neuron addition is not satisfied, then the learning algorithm adapts the parameters for the existing hidden neurons using a Hebbian like learning rule. Instead of updating all the weights, the learning rule updates only the synapses which caused the postsynaptic spike. This minimizes the delay between the spike time, for the correct hidden layer neuron, and the ideal firing time.

In the second stage of learning, the learning algorithm maps the response of most dominant hidden layer neurons in the hyperdimensional space to the output space. It identifies the dominant neurons by pruning the hidden layer neurons with low average rank order and then the output layer weights for the dominant hidden neurons are initialized using rank order learning. Thereafter, the output layer weights are tuned using a similar Hebbian like learning rule for accurately mapping the hidden layer response to the output space. In this case also, only the synapses which caused the postsynaptic spike are updated.

The performance of the GPSNN was compared with existing spiking neural networks with a fixed architecture as well as those with an evolving architecture, on benchmark datasets, in two different experiments. In the first experiment, the performance of the GPSNN was compared with other batch learning algorithms, namely SpikeProp [4], MuSpiNN [18], Multi-spike learning [19] and synaptic weight association training [9]. In the second experiment, the performance of the GPSNN was compared with an evolving spiking neural network, namely OSNN. The repeated two-way ANOVA was also used to statistically validate the results of performance comparison between the GPSNN and OSNN. The results clearly highlight that the GPSNN can achieve better results than other spiking neural networks and at the same time uses a compact network structure which requires considerably lesser computational effort.

The rest of the paper is organized as follows. Section II describes the architecture and the two stage learning algorithm for the GPSNN. Section III evaluates the performance of the GPSNN and highlights the results of a comparison with the performance of other spiking neural networks. Finally, section IV summarizes the conclusions from this paper.

II. Growing-pruning spiking neural network

In this section, we describe the architecture of the Growing-Pruning Spiking Neural Network (GPSNN) and its learning algorithm for the training samples, $\{(\mathbf{x_1}, c_1), (\mathbf{x_2}, c_2), \cdots, (\mathbf{x_d}, c_d), \cdots, (\mathbf{x_D}, c_D)\}$. Here, $\mathbf{x_d}$ is the M-dimensional feature vector $[x_d^1, \cdots, x_d^i, \cdots, x_d^M]$ and $c_d \in [1, 2, \cdots, N]$ is the class label, N being the number of classes. The objective of the learning algorithm is to closely approximate the functional mapping between an input sample and its class label.

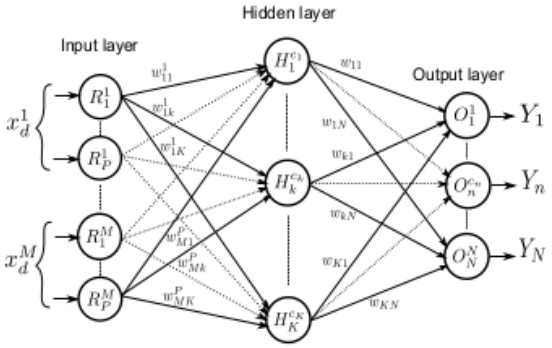

Fig. 1. Architecture of the GPSNN

A. Architecture of the GPSNN

The GPSNN is a fully connected three-layered network, as shown in Figure 1. The input layer consists of Gaussian receptive field neurons that convert each input sample into a spike pattern using population coding [20]. The hidden layer neurons and the output layer neurons are spiking neurons modeled as leaky integrate-and-fire [17] neurons. It may be noted that the number of neurons in the hidden layer is grown/adapted by the learning algorithm during training, whereas that in the output layer is fixed and equal to the number of classes. The predicted class for any sample is determined by the output neuron with minimum spike latency. Next, we will describe the receptive field neuron and the spiking neuron in detail. Without loss of generality, let us assume that $(\mathbf{x_d}, c_d)$ is the current training sample and the hidden layer has grown to K neurons after presentation of $d-1$ training samples.

Receptive field neuron: The receptive field neurons employ a modified population coding scheme to convert an input sample into a spike pattern. The modified population coding does not require multiple delays for each receptive field resulting in a compact network structure, consequently lesser computational effort is required. For any given sample, i^{th} feature is encoded using P receptive field neurons, $[R_1^i, \cdots, R_h^i, \cdots, R_P^i]$, with spike times in the interval $[0, T]$, where T is the simulation interval [4]. Similar to OSNN [11], a value of 10 ms has been used for T in this paper. The spike time, t_h^i, for a receptive field neuron is given as:

$$t_h^i = T(1 - \phi_h^i) \tag{1}$$

where ϕ_h^i is firing strength for the receptive field neuron

$$\phi_h^i = exp\left(-\frac{\left(x_d^i - \mu_h^i\right)^2}{2\sigma^2}\right) \tag{2}$$

where μ_h^i and σ are initialized as

$$\mu_h^i = I_{min}^i + \frac{(2h-3)(I_{max}^i - I_{min}^i)}{2(P-2)} \tag{3}$$

$$\sigma = \frac{(I_{max}^i - I_{min}^i)}{\gamma(P-2)} \tag{4}$$

where $[I_{min}^i, I_{max}^i]$ is the range of the i^{th} feature. The parameter γ controls the overlap between the receptive fields. If the value of γ is set greater than one then there is little overlap between receptive fields, as a result, the spike time for most of the receptive fields will be close to T. On the other hand, if the value of γ is set too low, then there will be too much overlap between the receptive fields, causing most of the receptive fields to fire close to zero. In both the scenarios, most of the receptive fields will be firing very close to each other, which diminishes the ability of the network to distinguish between samples. Hence, the value of γ has been set to 0.7 for simulations in this paper.

Spiking neuron: The hidden layer and the output layer consist of leaky integrate-and-fire [17] spiking neurons. The response of the output neuron associated with class c_n, $O_n^{c_n}$, at time t is given by

$$Y_n(t) = \sum_{t_n} \delta(t - t_n) \tag{5}$$

$$t_n = \{t | V_n(t) > \theta_n\} \tag{6}$$

$$\lim_{t \to t_n, t > t_n} V_n(t) = 0 \tag{7}$$

where δ is the Dirac delta function, θ_n is the threshold and t_n is the spike time of the neuron. $V_n(t)$ is the postsynaptic potential for the neuron at time t, defined as

$$V_n(t) = \int_0^t \sum_{k, t_k < t} w_{kn} \epsilon(t - t_k) dt \tag{8}$$

where w_{kn} is the weight of the synapse between k^{th} hidden layer neuron, $H_k^{c_k}$ and $O_n^{c_n}$. t_k is the spike time for $H_k^{c_k}$ and c_k is its associated class. ϵ is the normalized potential for the leaky integrate-and-fire neuron, realized using spike-response function [4], given as

$$\epsilon(s) = \frac{s}{\tau} exp\left(1 - \frac{s}{\tau}\right) \tag{9}$$

where τ is the membrane time constant for the neuron. For simulation study, it has been set at 2 ms. Since, the learning algorithm uses only the first spike generated by the hidden layer and the output layer neurons, t_k and t_n will represent the first spike by respective neurons.

B. Two stage learning algorithm of GPSNN

The learning algorithm of the GPSNN employs a two stage learning mechanism. The first stage, called structure learning involves growing/adapting the hidden layer neurons and the second stage, called parameter learning, consists of pruning hidden layer neurons and adaptation of output neuron weights. The main advantage of the proposed two stage approach is that the desired output of the trained neurons is known in both the stages. Next, we will describe each learning stage in detail.

- **Structure learning:** In this stage, the learning algorithm maps input samples to a hyperdimensional space by growing and adapting the hidden layer neurons, depending on the information present in the training sample. We will explain both the learning strategies in detail.

  Neuron addition strategy : A new neuron is added, if the network does not contain the information present in the training sample. This is the case when all the hidden layer neurons fire close to the end of simulation interval. The criteria for neuron addition is given as

  $$\begin{aligned} &\textbf{if} \quad (c_d = c_f)\ \&\ ((t_f - T_0) > T_C) \\ &\textbf{then} \quad \text{add a neuron} \end{aligned} \tag{10}$$

  where t_f and c_f are, respectively, the spike time and the associated class for the hidden layer neuron with minimum spike latency. T_0 is the ideal firing time and T_C is the neuron addition threshold in case the winner neuron is associated with the correct class. In this paper, T_0 has

been set to $T/2$. If the winner neuron is not associated with the correct class, the criteria for neuron addition is

$$\begin{aligned} &\textbf{if} \quad (c_d \neq c_f)\ \&\ ((t_f - T_0) > T_M) \\ &\textbf{then} \quad \text{add a neuron} \end{aligned} \tag{11}$$

where T_M is the neuron addition threshold in case winner neuron is associated with the wrong class. The use of temporal thresholds i.e. T_C and T_M, and class-specific information present in the network provides an accurate measure of the information present in the sample with regards to the network.

The weights and threshold for the newly added neuron are initialized such that the information present in the training sample is accurately stored in the network. An improved rank order coding [12] technique is used to initialize the weights, since it is an effective learning technique for one-shot learning mode. In rank order coding [10], as the number of features increase, some synapse become unresponsive due to low weights. To avoid this problem, in GPSNN, weights are initialized by using rank order coding across receptive fields encoding a single feature which improves the separability of the input samples. The weight for the connection between R_h^i and $H_{(K+1)}^{c_{(K+1)}}$ is given as:

$$w_{i(K+1)}^h = \lambda^{r_h^i} \tag{12}$$

where λ is the modulation factor and r_h^i is the rank of the receptive field. λ controls the reduction in PSP contribution for subsequent spikes and is chosen between zero and one. If it is set close to one then there is little difference in the PSP contribution for subsequent spikes. In case it is set close to zero then the PSP contribution for subsequent spikes decays quickly. In both the cases, the ability of the network to distinguish between samples is diminished. Therefore, in this paper, λ is set to 0.8. The rank of the receptive field is given by

$$r_h^i = \sum_{y=1, y\neq h}^{P} F_R(h, y) \tag{13}$$

where F_R is the ranking function, given as

$$F_R(x, y) = \begin{cases} 0 & \phi_x^i \geq \phi_y^i \\ 1 & otherwise \end{cases} \tag{14}$$

The threshold for the newly added neuron is initialized using it's PSP, for the sample $(\mathbf{x}_d, c_d)$, at the ideal firing time T_0. Thus, the threshold for the neuron is given as

$$\theta_{(K+1)} = \int_0^{T_0} \sum_{i,h,t_h^i < T_0} w_{i(K+1)}^h \epsilon(t - t_h^i) dt \tag{15}$$

This ensures that the newly added neuron will fire precisely at T_0 for the current sample.

Parameter update strategy : When the criteria for neuron addition is not satisfied, the weights for the correct class neuron (fc) with minimum spike latency are updated according to a Hebbian like learning rule. The weight for the presynaptic synapses with lower spike latency then fc is adapted according to the difference between existing weight, which represents the rank of previously seen samples, and the rank of the current sample. This approach helps to minimize the difference in the spike time of the hidden neuron and ideal firing time. The learning rule is given as

$$w_{v(fc)}^u = w_{v(fc)}^u + \eta(\lambda^{r_u^v} - w_{v(fc)}^u) \quad \forall (u, v) \in \psi(fc) \tag{16}$$

where η is the learning rate and is set to 0.001. $\psi(fc)$ represent the set of synapses with lower spike latency, given by

$$\psi(fc) = \{(u, v) | (t_u^v < t_{(fc)})\ \&\ (w_{u(fc)}^v < \lambda^{r_u^v})\} \tag{17}$$

Once the hidden layer has evolved, the network enters the parameter learning stage to estimate the weights between the most dominant hidden layer neurons and the output layer neurons.

- **Parameter learning :** After the structure of the network has been determined, the parameters for the most dominant hidden neurons are initialized using average class-wise rank order. The class-wise dominance of a k^{th} hidden layer neuron (g_k) is estimated as:

$$g_k = \frac{1}{D_{c_k}} \sum_{d, c_d = c_k} r_k \tag{18}$$

where D_{c_k} is the number of samples with class label c_k and r_k is the rank of the k^{th} hidden layer neuron (determined using Equation (14),(13)). A low value for class-wise dominance suggests that the neuron fires very late for most of the training samples, which indicates that the it's spike time is more than the spike time for output neuron with minimum spike latency for most samples. Such a neuron $(g_k < \beta_R)$ has minimal contribution to the network output and is pruned by initializing the weights to zero. Hence, the weights for the neurons with a given class-wise dominance are initialized as follows.

$$w_{kn} = \begin{cases} \frac{1}{D_{c_n}} \sum_{d, c_d = c_n} r_k & g_k \geq \beta_R \\ 0 & g_k < \beta_R \end{cases} \tag{19}$$

where β_R is the neuron pruning threshold. For the simulation in this paper β_R was set to 0.3. Once, the output neuron weights have been initialized, the output neuron weights are adjusted only in case of misclassification, to make the correct output neuron fire early, using a Hebbian like rule given by

$$w_{uv} = w_{uv} + \eta(\lambda^{r_u} - w_{uv}) \quad \forall \quad (u, v) \in \psi(c_r) \tag{20}$$

where

$$\psi(c_r) = \{(u, v) | (c_v = c_r)\ \&\ (c_u = c_r)\ \&\ (\lambda^{r_u} > w_{uv})\} \tag{21}$$

It should be noted that the hidden neuron weights are not adjusted in this stage. Further, the output layer weights for the pruned neurons are also not adjusted.

Algorithm 1 Two stage learning algorithm for the GPSNN

Structure Learning:
for $d \leftarrow 1$ to D **do**
  $f \leftarrow$ Hidden neuron with minimum spike latency
  if $(c_d = c_f)$ & $((t_f - T_0) > T_C)$ **then**
    Add a neuron
  else if $(c_d \neq c_f)$ & $((t_f - T_0) > T_M)$ **then**
    Add a neuron
  else
    Update the f^{th} neuron
  end if
end for
Parameter Learning:
while Training accuracy is less than a pre-defined value **do**
  for $i \leftarrow 1$ to D **do**
    if Predicted class $\neq$ Actual class **then**
      Update the correct class neuron
    end if
  end for
end while

The parameter learning stage of training phase is repeated for every training sample. The training phase is stopped when a given training accuracy is achieved or a maximum number of epochs have been reached.

The learning algorithm for the GPSNN has been provided in a pseudocode format in Algorithm 1.

III. Performance Evaluation of GPSNN

In this section, the performance of the GPSNN is evaluated and compared with other spiking neural networks, in two different studies. In the first study, the performance of the GPSNN is compared with the performance of other well-known batch learning algorithm for the Fisher Iris and Wisconsin breast cancer problems. In the second study, the performance of the GPSNN is compared with the performance of Online Spiking Neural Network (OSNN) for five benchmark datasets from the UCI [21] machine elarning repository. It may be noted that OSNN was used for comparison as it is the only evolving spiking neural network which can conveniently handle problems with real valued data. The performance comparison has been done on three metrics namely, overall classification accuracy, number of network parameters and the number of epochs needed for convergence. The statistical test of repeated two-way ANOVA has also been conducted to validate the results of performance comparison between the GPSNN and OSNN.

All the simulations were carried out on MATLAB 2013b in windows environment with 16GB RAM.

A. Performance comparison with batch learning algorithms

In this section, the performance of the GPSNN is evaluated and compared with the performance of other well-known batch learning algorithms in spiking neural network literature, namely, SpikeProp [4], MuSpiNN [18], Multi-spike learning [19] and Synaptic Weight Association Training [9] (SWAT). The Fisher Iris and Wisconsin breast cancer problems have been used for the comparison as these are the only problems for which the aforementioned algorithms have been evaluated. The Fisher Iris problem is a three class problem with 150 samples and four features per sample. The Wisconsin breast cancer problem is a binary problem with 699 samples and nine features per sample. The performance evaluation for both the problems was done by using 10 samples per class for training.

The results of performance evaluation are shown in Table I. It may be noted that the results for MuSpiNN and Multi-spike learning have been reproduced from [19] and for SpikeProp have been reproduced from [5]. The results for SWAT were reproduced from [9] and the results for the GPSNN were generated on 10 random trials using the setting in [19]. The table also shows the architecture, number of network parameters and the number of epochs for each algorithm. The architecture for each algorithm has been shown in the format $n_i - n_h - n_o$ where n_i represent the number of neurons used for encoding real valued input; and n_h, n_o represent the number of hidden neurons and output neurons respectively. The number of network parameters was estimated as follows

$$\text{No. of network parameters} = n_i * n_f * n_l * n_h + n_h * n_o \quad (22)$$

where n_f and n_l are the number of receptive fields and delays respectively. It may be noted that the number of epochs for GPSNN is only applicable to the parameter learning phase. The structure learning phase takes only single epoch.

It can be observed from that table that, for Fisher Iris problem, the performance of the GPSNN is 1-4% better than the other algorithms using a quarter of the parameters used by the next best performing algorithm. Similarly, for the Wisconsin breast cancer problem, it can achieve similar

TABLE I
Performance comparison of the GPSNN for Fisher Iris and Wisconsin breast cancer problems

Algorithm	Fisher Iris problem				Wisonsin breast cancer			
	Architecture[a]	# Network Parameters	Testing	# Epochs	Architecture[a]	# Network Parameters	Testing	# Epochs
SpikeProp	17-8-1	2304	92.7	37	-	-	-	-
MuSpiNN	17-8-1	576	95.87	192	46-8-1	1880	94.66	166
Multi-spike (GMES)	21-8-1	880	94.72	241	46-8-1	1880	95.32	209
SWAT	16-208-3	624	95.3(3.6)	500	9-117-2	234	96.7(2.3)	500
GPSNN	20-6-3	138	96.(1.5)	97	45-2-2	94	96.3(0.5)	123

[a] Input neurons represent the total number of neurons used for encoding.

performance using half the number of parameters. This is due to the modified population coding scheme used by the GPSNN which reduces the number of parameters considerably, and hence minimizing the computational effort required.

It can also be observed from the Table I that the GPSNN requires fewer epochs in comparison to other algorithm for achieving comparable performance. This is because of the improved rank order learning used for initializing weights which provides faster convergence.

B. Performance comparison with OSNN

In this section, the performance of the GPSNN is evaluated and compared with the performance of Online Spiking Neural Network (OSNN) [11] for five benchmark data sets from the UCI [21] machine learning repository. Table II shows the details of the data sets used for comparison.

TABLE II
SPECIFICATION OF THE UCI BENCHMARK DATA SETS USED FOR COMPARISON

Data set	# Features	# Classes	# Samples	
			Training	Testing
Iris	4	3	75	75
Breast cancer (BC)	9	2	350	333
Pima Diabetes (PD)	9	2	384	384
Liver disorders (LD)	6	2	170	175
Ionosphere (ION)	34	2	175	176

The results for OSNN have been reproduced from [11] and those for the GPSNN were generated using 10 random trials. It should be noted that the results for the GPSNN were generated using the same parameter setting as in OSNN for all the data sets except Iris problem. For Iris problem, OSNN employs different number of receptive fields, for different features, whereas results for the GPSNN were generated using five receptive fields for all features.

TABLE III
PERFORMANCE COMPARISON OF GPSNN WITH OSNN - UCI BENCHMARK DATA SETS

Data set	OSNN		Two stage GPSNN		
	Training (%)	Testing (%)	Training (%)	Testing (%)	# Epochs
Iris	87.2(4.1)	86.1(6.7)	97.48(1.16)	96.59(1.55)	92
BC	91.1(2.0)	90.4(1.8)	96.63(0.61)	96.4(0.9)	103
PD	68.2(2.0)	63.5(3.0)	72.6(2.47)	71.61(2)	51
LD	58.7(2.2)	56.66(1.8)	61.18(2.73)	59.79(2.62)	105
ION	76.7(2.4)	76.6(4.8)	89.2(2.36)	88.01(1.93)	121

Table III shows the result of performance evaluation of OSNN and the GPSNN. It can be observed that the GPSNN performs 5-9% better than OSNN for simple problems like Iris flower classification and Wisconsin breast cancer. For difficult problems like Pima diabetes and liver disorders the performance of the GPSNN is about 3-8% better than OSNN. For a problem with large number of features like Ionosphere, the performance of the GPSNN is about 12% better than OSNN.

TABLE IV
COMPARISON OF NUMBER OF NETWORK PARAMETERS FOR THE GPSNN AND OSNN [11]

Data set	# Neurons		# Network Parameters	
	OSNN	Two stage GPSNN	OSNN	Two stage GPSNN
Iris	7-21	6-10	5390-16191	138-230
BC	10-16	7-11	8660-13856	392-616
PD	8-18	7-11	6160-13860	392-616
LIV	4-7	6-8	2312-4046	228-304
ION	4-11	7-10	12680-34870	1442-2060

Table IV shows the number of neurons and the number of network parameters used by OSNN and the GPSNN. The number of network parameters for OSNN were calculated using the Equation (22).It may be noted that, for the GPSNN, n_h is the number of dominant neurons and the number of delays is equal to one. From the table, it can be clearly observed that the modified population coding results in considerable lesser network parameters.

1) Statistical Comparison: In order to statistically validate the performance of the GPSNN in comparison to OSNN, we conducted the two-way ANOVA test. The two-way ANOVA test measures whether the population means for the various factors are equivalent (null hypothesis). If the p-value for the computed F-statistic is less than 0.05 (95% confidence interval) then one rejects the null hypothesis.

In this study, the mean and variance for the GPSNN and OSNN were obtained by 10 random trials for five different data sets. The performance results of the GPSNN and OSNN represent two different groups and ANOVA monitors three different kinds of variations within the data i.e. within group variations, between group variations and the overall variation. The F-statistic obtained for the reported results was 10.47 which is greater than the critical F-statistic at 95% confidence interval ($F_{1,4,0.05}$ is 7.71). Hence, one can reject the null hypothesis at 95% confidence interval. Since, there are only two classifiers used for comparison, we can say that the performance of the GPSNN is significantly better than OSNN.

IV. CONCLUSION

In this paper, a new three-layered spiking neural classifier, the GPSNN, for pattern classification was presented. In the input layer, a modified population coding scheme is employed to convert a real valued input into a spike pattern which requires lesser computational effort due to lower number of network parameters. The hidden and output layers consist of leaky integrate-and-fire spiking neurons. The learning algorithm of the proposed approach employs a two stage learning mechanism. In the first stage, the hidden layer neurons are grown/adapted according to a temporal similarity measure and class-wise information present in the network. The weight adaptation employs a Hebbian like learning rule to make the correct hidden neurons fire early. In the second stage, the least responsive neurons are pruned and the weights between the most dominant neurons and output neurons are adapted

according a similar learning rule. The performance of the GPSNN was evaluated and compared with the performance of well known spiking neural networks, with a fixed architecture, for benchmark data sets. A comparison was also done with an evolving spiking neural network and the results of performance comparison highlight that the GPSNN performs significantly better than the existing spiking neural networks. It can also be observed from the results that the GPSNN achieves the superior performance using a compact network structure.

REFERENCES

[1] R. V. Rullen and S. J. Thorpe, "Rate coding versus temporal order coding: what the retinal ganglioncells tell the visual cortex," *Neural computation*, vol. 13, no. 6, pp. 1255–1283, 2001.

[2] R. V. Rullen, R. Guyonneau, and S. J. Thorpe, "Spike times make sense," *Trends in Neuroscience*, vol. 28, no. 1, pp. 1–4, 2005.

[3] D. A. Butts, C. Weng, J. Jin, C. Yeh, N. A. Lesica, J. Alonso, and G. B. Stanley, "Temporal precision in the neural code and the timescales of natural vision," *Nature*, vol. 449, no. 7158, pp. 92–95, 2007.

[4] S. M. Bohte, J. N. Kok, and H. L. Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.

[5] S. Ghosh-Dastidar and H. Adeli, "Improved spiking neural network for EEG classification and epilepsy and seizure detection," *Integrated Computer-Aided Engineering*, vol. 14, no. 3, pp. 187–212, 2007.

[6] R. Gütig and H. Sompolinsky, "The tempotron a neuron that learns spike timing based decisions," *Nature Neuroscience*, vol. 9, no. 3, pp. 420–428, 2006.

[7] R. V. Florian, "The chronotron: A neuron that learns to fire temporally precise spike patterns," *PLoS ONE*, vol. 7, no. 8, p. e40233, 2012.

[8] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: Sequence learning, classification and spike shifting," *Neural Computation*, vol. 22, no. 2, pp. 467–510, 2010.

[9] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: A Spiking Neural Network Training Algorithm for Classification Problems," *IEEE Transactions on Neural Networks*, vol. 21, no. 11, pp. 1817–1830, 2010.

[10] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Fast and adaptive networks of spiking neurons for multi-view visual pattern recognition," *Neurocomputing*, vol. 71, no. 13, pp. 2563–2575, 2008.

[11] J. Wang, A. Belatreche, L. Maguire, and T. M. McGinnity, "An online supervised learning method for spiking neural networks with adaptive structure," *Neurocomputing*, vol. 144, no. 0, pp. 526–536, 2014.

[12] S. Thorpe and J. Gautrais, "Rank order coding," in *Computational Neuroscience*. Springer, 1998, pp. 113–118.

[13] S. J. Thorpe and M. Imbert, "Biological constraints on connectionist modelling," *Connectionism in perspective*, pp. 63–92, 1989.

[14] S. Suresh, N. Sundararajan, and P. Saratchandran, "Risk sensitive loss functions for sparse multi-category classification problems," *Information Sciences*, vol. 179, no. 21, pp. 2621–2638, 2008.

[15] S. Dora, S. Ramasamy, and S. Sundaram, "A basis coupled evolving spiking neural network with afferent input neurons," in *Proceedings of the International joint conference on neural networks*, 2013, pp. 1–8.

[16] S. Dora, S. Sundaram, and N. Sundararajan, "A Sequential Learning Algorithm for a Minimal Spiking Neural Network (MSNN) Classifier," in *Proceedings of the International Joint Conference on Neural Networks*, 2014, pp. 2415–2421.

[17] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain Research Bulletin*, vol. 50, pp. 303–304, 1999.

[18] S. Ghosh-Dastidar and H. Adeli, "A new supervised learning algorithm for multiple spiking neural network with application in epilepsy and seizure detection," *Neural Networks*, vol. 22, no. 10, pp. 1419–1431, 2009.

[19] Y. Xu, X. Zeng, L. Han, and J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Networks*, vol. 43, pp. 99–113, 2013.

[20] S. Bohte, H. L. Poutré, and J. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multi-layer RBF networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 426–435, 2002.

[21] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml